Create a repository on github named "python-test" and send us the link. you will upload your work to this repo.

Dataset selection:

Use: https://github.com/swapniljariwala/nsepy

Source OCLHV data for NSE stocks (INFY,TCS) between 2015-2016. Data level - Daily.

Source OCLHV data for NIFTY IT index. Data level - Daily.

Part 1:

1. Create 4,16,....,52 week moving average(closing price) for each stock and index. This should happen through a function.

2. Create rolling window of size 10 on each stock/index. Handle unequal time series due to stock market holidays. You should look to increase your rolling window size to 75 and see how the data looks like. Remember they will create stress on your laptop RAM load. ( Documentation you might need: http://in.mathworks.com/help/econ/rolling-window-estimation-of-state-space-models.html)

3. Create the following dummy time series:

   3.1 Volume shocks - If volume traded is 10% higher/lower than previous day - make a 0/1 boolean time series for shock, 0/1 dummy-coded time series for direction of shock.

   3.2 Price shocks - If closing price at T vs T+1 has a difference > 2%, then 0/1 boolean time series for shock, 0/1 dummy-coded time series for direction of shock.

   3.3 Pricing black swan - If closing price at T vs T+1 has a difference > 2%, then 0/1 boolean time series for shock, 0/1 dummy-coded time series for direction of shock.

   3.4 Pricing shock without volume shock - based on points a & b - Make a 0/1 dummy time series.

Part 2 (data visualization ):

For this section, you can use only bokeh. https://bokeh.pydata.org/en/latest/docs/gallery.html

1. Create timeseries plot of close prices of stocks/indices with the following features:

2. Color timeseries in simple blue color.

3. Color timeseries between two volume shocks in a different color (Red)

4. Gradient color in blue spectrum based on difference of 52 week moving average.

5. Mark closing Pricing shock without volume shock to identify volumeless price movement.

6. Hand craft partial autocorrelation plot for each stock/index on upto all lookbacks on bokeh - sample reference - https://www.statsmodels.org/dev/generated/statsmodels.graphics.tsaplots.plot\_pacf.html

Part 3 (data modelling - optional)

For this section, you should use sklearn.

1. Quick build any two models. Quick build is defined as grid search of less than 9 permutation combinations. You can choose the two options of multiple multivariate models from those mentioned below. The goal is to to predict INFY,TCS prices for tomorrow.  Models that you can choose:

    1.1 http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LassoLars.html#sklearn.linear\_model.LassoLars

    1.2 http://scikit-learn.org/stable/modules/generated/sklearn.linear\_model.LinearRegression.html#sklearn.linear\_model.LinearRegression

    1.3 http://scikit-learn.org/stable/modules/linear\_model.html#ridge-regression

    1.4 http://scikit-learn.org/stable/modules/svm.html#regression

    1.5 http://scikit-learn.org/stable/modules/ensemble.html#regression

2. Write testcases for the two models you have chose. Your testing should take atleast 5 time steps except today . your testcases must written using PYTEST.

3. Prove your model does not violate any basic assumption. To understand "model assumptions", read

https://www.albert.io/blog/key-assumptions-of-ols-econometrics-review/

4. Select best performing model, and tune it - Demonstrate that your tuning has resulted in a clear difference between quick build and tuning.

5. Your submission will be judged on MAPE of testing data.

6. Extra credit - Nest a model to predict volume shock into your time series model - same conditions applied as above.

7. Extra extra credit - Create a bare python file in the following fashion “ python stockpredictor.py ‘INFY’ “ should return prediction in less than 100 ms.